**Universitatea Tehnică „Gheorghe Asachi” din Iași**

**Facultatea de Automatică și Calculatoare**

**Domeniul: Calculatoare și Tehnologia Informației**

# Detecția și recunoașterea elementelor grafice într-o fereastră.

Proiect la disciplina

Prelucrarea Imaginilor

**Echipa E6:** Petrica Petru, Enachi Vasile

# Capitolul 1. Introducere

Interfața grafică cu utilizatorul (GUI) este un element crucial în zilele noastre și este foarte important pentru a face interacțiunea dintre calculatoare și utilizatori ușoară.

Detecția și recunoașterea elementelor grafice într-o interfață grafică poate servi pentru o potențială automatizare a lucrului cu acestea fără a fi necesară intervenția unui om. Un exemplu în acest sens ar fi generarea codului având o imagine cu interfața grafică, deși acest câmp de studiu rămâne unul puțin explorat.

De asemenea, o astfel de aplicație ar putea fi folosită ulterior pentru a oferi posibilitatea de a interacționa la rândul lor cu un GUI persoanelor ce suferă de afecțiuni ale vederii, pentru care informația despre interfața poate fi transmisă în alt mod.

De-a lungul acestui proiect ne propunem detecția și recunoașterea elementelor grafice dintr-o interfață grafică. Pentru sporirea probabilității de detecție a elementelor grafice, imaginea analizată nu trebuie să fie blurată sau să conțină zgomot.

## Obiective atinse în cadrul proiectului

- detectarea butoanelor
- detectarea text field-urilor
- detectarea radial buttons
- detectarea check – boxurilor și a stării acestuia (checked, unchecked)
- reprezentarea rezultatelor într-un format json
- tool de vizualizare a rezultatelor

## Metode existente

Într-un document de cercetare denumit “Extraction and Classification of User Interface Components from an Image” de Saad Hassan, Manan Arya , Ujjwal Bhardwaj ,Silica Kole aceștia propun un algoritm de determinare a componentelor GUI și anume:

- Detectare textului din imagine (coordonate dreptunghi-uri)
- Mascarea textului (pentru eliminarea detectării redundante)
- Detectarea componentelor (folosind o rețea neuronală MobileNet pentru detecția componentelor)
- Antrenarea modelului de detecție a componenetelor grafice
- Salvarea datelor componentelor extrase

În acest proiect ne propunem să elaborăm propriile funcții de detecție a componentelor grafice, având la dispoziție doar rețeaua preantrenată EAST(An efficient and Accurate Scene Text Detector) pentru identificarea regiunilor ce conțin text și motorul de identificare a textului Tesseract.

De precizat că atât rețeaua cât și motorul de identificare a textului nu sunt acurate.

# Capitolul 2. Tehnologii folosite pentru front-end

Pentru crearea tool-ului de vizualizare a fost folosită librăria tkinter din Python.

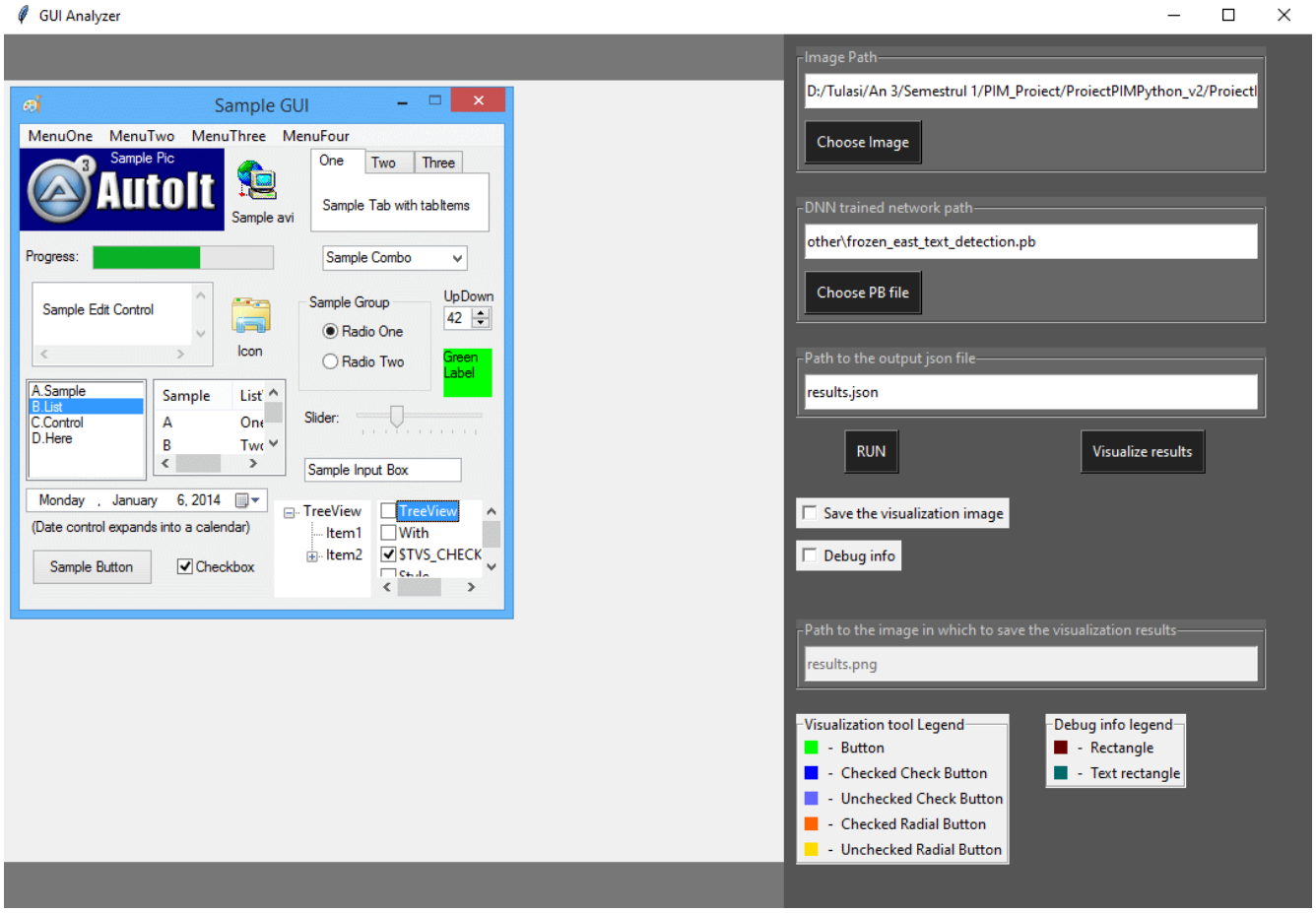



**Fig. 2.1. Interfața grafică a aplicației**

Elementele grafice (widget-urile) folosite în interfața grafică sunt:

- **Label** (pentru setarea fieldurilor de text)(ex. Legendă)
- **LabelFrame** (pentru încadrarea unor elemente grafice) (total 6)
- **Entry** (pentru introducerea path-ului imaginii sursă, rețelei neuronale ș.a.)
- **Canvas** (pentru afișarea imaginii sursă)
- **Button** (pentru apelarea unor funcții de callback corespunzătoare) (exemplu alegere path image, rețea, rulare analiză imagine sursă, vizualizare rezultate)
- **CheckBox** (pentru activarea opțiunilor de salvare a rezultatelor într-o imagine sau afișarea textului detectat)
- **Frame** (pentru încadrarea elementelor grafice, pentru crearea simbolurilor din legendă)

# Capitolul 3. Implementarea funcțiilor de detecție a elementelor grafice

## 3.1. Detectarea Dreptunghiurilor

Pașii pentru detectarea dreptunghiurilor:

1) Găsirea contururilor folosind funcția din opencv **findContours** cu parametrul de aproximație CHAIN_APPROX_SIMPLE pentru a identifica doar punctele semnificative
2) Aproximarea contururilor cu un poligon folosind funcția **approxPolyDP** și filtrând toate poligoanele care nu au 4 laturi
3) Filtrarea poligoanelor care nu sânt paralelograme (verificare prin program că laturile opuse sunt paralele)
4) Filtrarea paralelogramelor care nu sunt dreptunghiuri (verificarea prin program ca diferența coordonatelor asociate laturilor opuse să fie mai mică ca 0.1 * lungimea laturii corespunzătoare)
5) Aproximarea poligonului cu un dreptunghi folosind **boundingRect**
6) Eliminarea dreptunghiurilor cu o arie mai mică decât o arie specificată
7) Eliminarea valorilor redundante

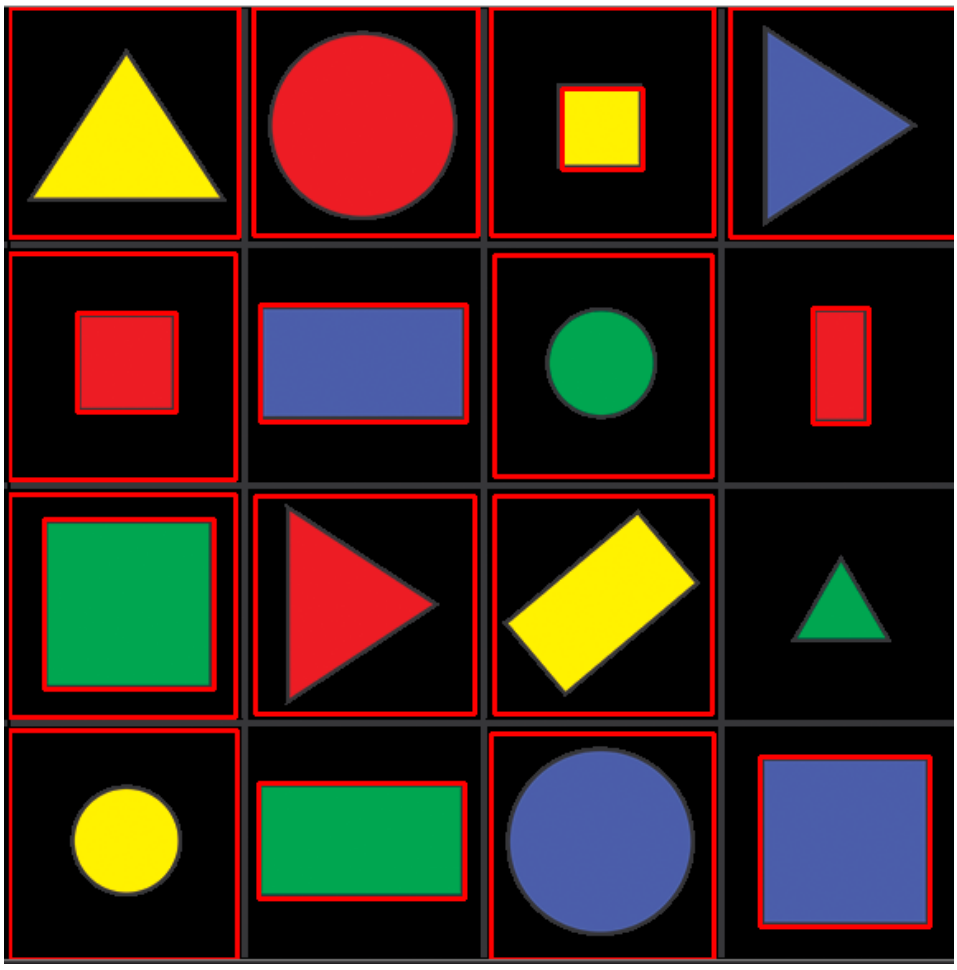


**Fig. 3.1. Detectarea dreptunghiurilor într-o imagine**

În figura 3.1 este prezentată un exemplu de aplicare a funcției de detecție dreptunghiuri, folosind valorea 966 pentru area minimă impusă, pentru o imagine de figuri.

## 3.2. Detectarea regiunilor ce conțin text (EAST neural network)

Pașii pentru detectarea textului:

1) redimensionarea imaginii așa încât lungimea și lățimea să fie multiplu de 32 (EAST detector accepta doar astfel de imagini)
2) preprocesarea imaginii pentru clasificarea ei în rețeaua neuronală folosind funcția **blobFromImage** (scăderea mediei rețelei neuronale(123.68, 116.78, 103.94) și scalarea imaginii)
3) transmiterea straturilor obținute către rețea și obținerea regiunilor cu text și a unor probabilități de prezență a textului în regiunile respective
4) decodarea predicțiilor folosind probabilitatea minimă de 10%
5) aplicarea funcției **non_max_supression**(imutils.object_detection) asupra bounding box-urilor pentru a înlătura suprapunerea lor
6) rescalarea la dimensiunile originale

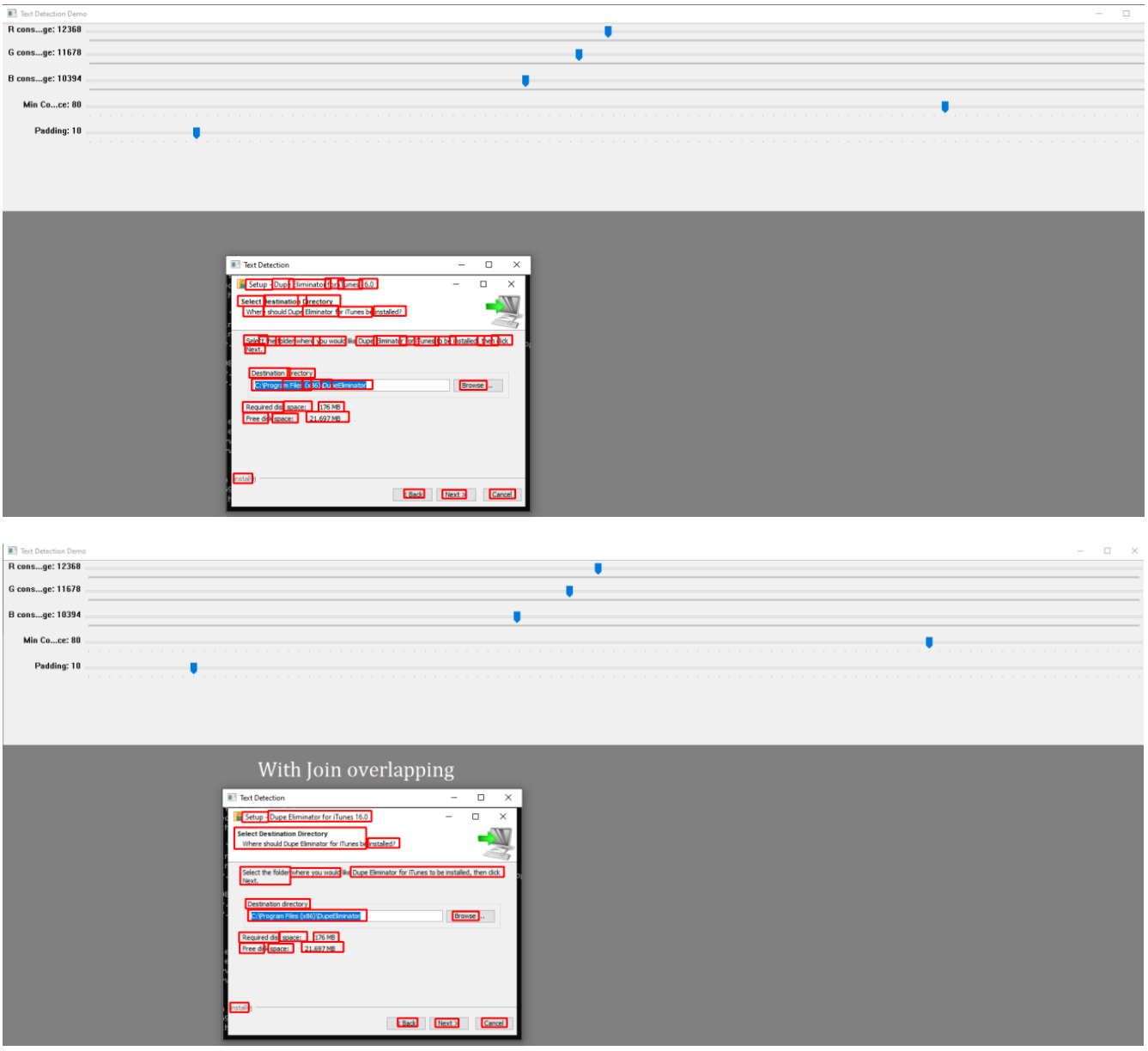



**Fig. 3.2. Detectarea regiunilor de text într-o imagine**

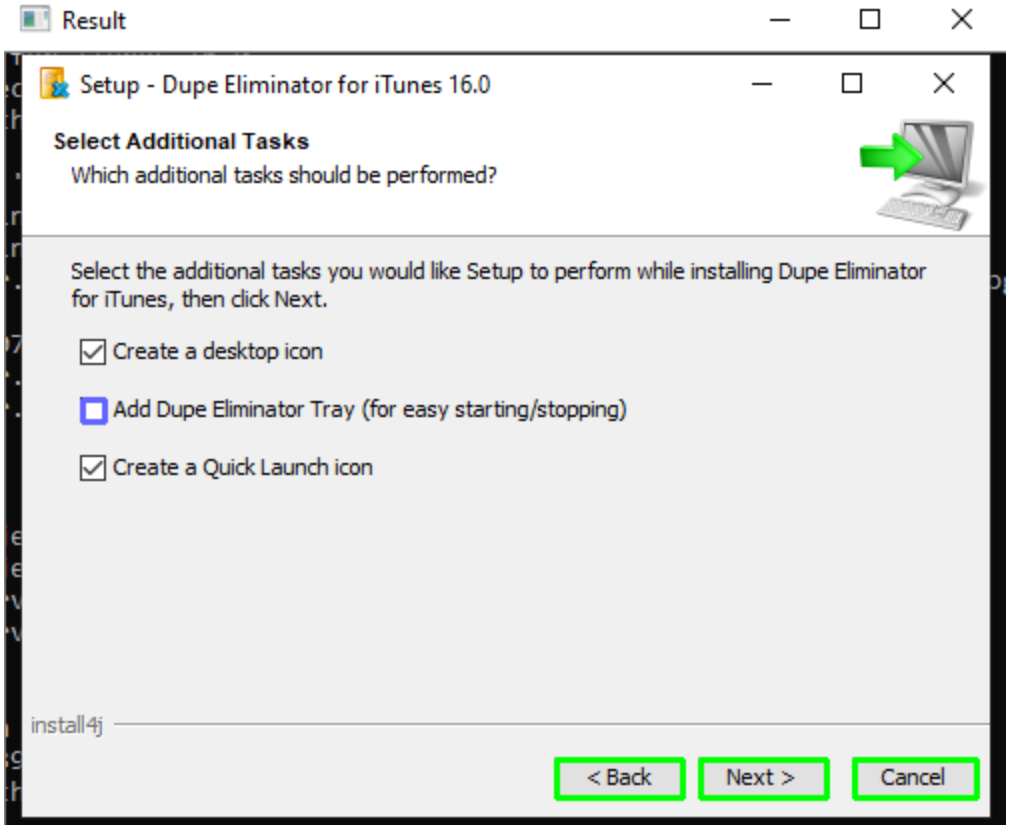
## 3.3. Detectarea Butoanelor

Pentru detectarea butoanelor se folosesc rezultatele obținute din funcțiile anterioare (detectarea dreptunghiurilor din contururi și detectarea textului).

Pașii pentru detectarea butoanelor:

1) Înlăturarea dreptunghiurilor exterioare (dacă un dreptunghi D1 conține înăuntrul său un alt dreptunghi D2, atunci D1 este considerat dreptunghi exterior și este eliminate din mulțimea dreptunghiurilor)
2) Eliminarea dreptunghiurilor pentru care lungimea este mai mică ca înălțimea (butoanele reprezintă dreptunghiuri orizontale, adică w > h)
3) Impunerea condiției ca raportul dintre (dreptunghiul intersectat dintre dreptunghiul de contururi și cel de text) și (minimul dintre dreptunghiul de contururi și cel de text) să fie mai mare de 0.8. Această condiție exclude dreptunghiurile care nu se intersectează sau care se intersectează parțial.


4) Impunerea condiției ca diferența absolută dintre înălțimile celor 2 dreptunghiuri să fie mai mică ca jumătate din înălțimea maximă a lor (condiție ce forțează respectarea ca dreptunghiul de text să fie înăuntrul dreptunghiului de contururi)



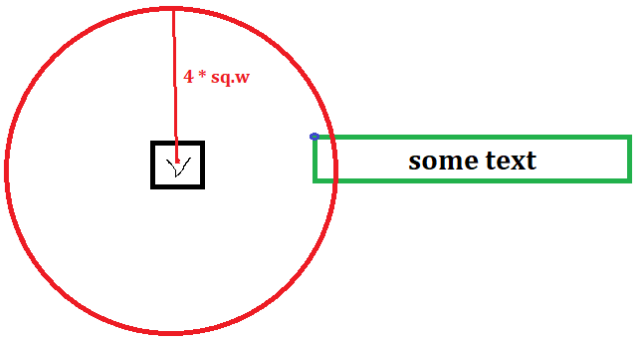

Butoanele sunt identificate cu conturul verde.

## 3.4. Detectarea Check-box-urilor

Pentru detectarea check-box-urilor se folosesc rezultatele obținute din funcțiile anterioare (detectarea dreptunghiurilor din contururi și detectarea textului).

Pașii pentru detectarea check-box-urilor :

1) Înlăturarea dreptunghiurilor de contururi pentru care diferența absolută dintre lungimea și lățimea dreptunghiului este mai mare de 20 pixeli (această condiție forțează ca dreptunghiul să fie aproximativ un pătrat)
2) Înlăturarea dreptunghiurilor de contururi pentru care aria dreptunghiului > 200 (pixeli$^2$) (prin această condiție impunem ca dreptunghiul să fie relativ de dimensiune mică)
3) Determinarea textului asociat check-boxului prin impunerea condițiilor:
   - ordonata dreptunghiului de text să fie la maxim 10 pixeli diferență de ordonata check-box-ului (condiție de optimizare a codului)
   - impunerea condiției ca ordonata dreptunghiului de text să fie conținută în cercul centrat în centrul check-boxului cu raza de 4 * checkbox.width




4) Determinarea state-ul check-box-ului (checked, unchecked) prin aplicarea metodei de binarizare **OTSU** și verificarea ca pixelii de culoarea neagră să reprezinte mai mult de 20 % din aria check-box-ului (funcționează pentru intensitate mai închisă a bifului față de background)

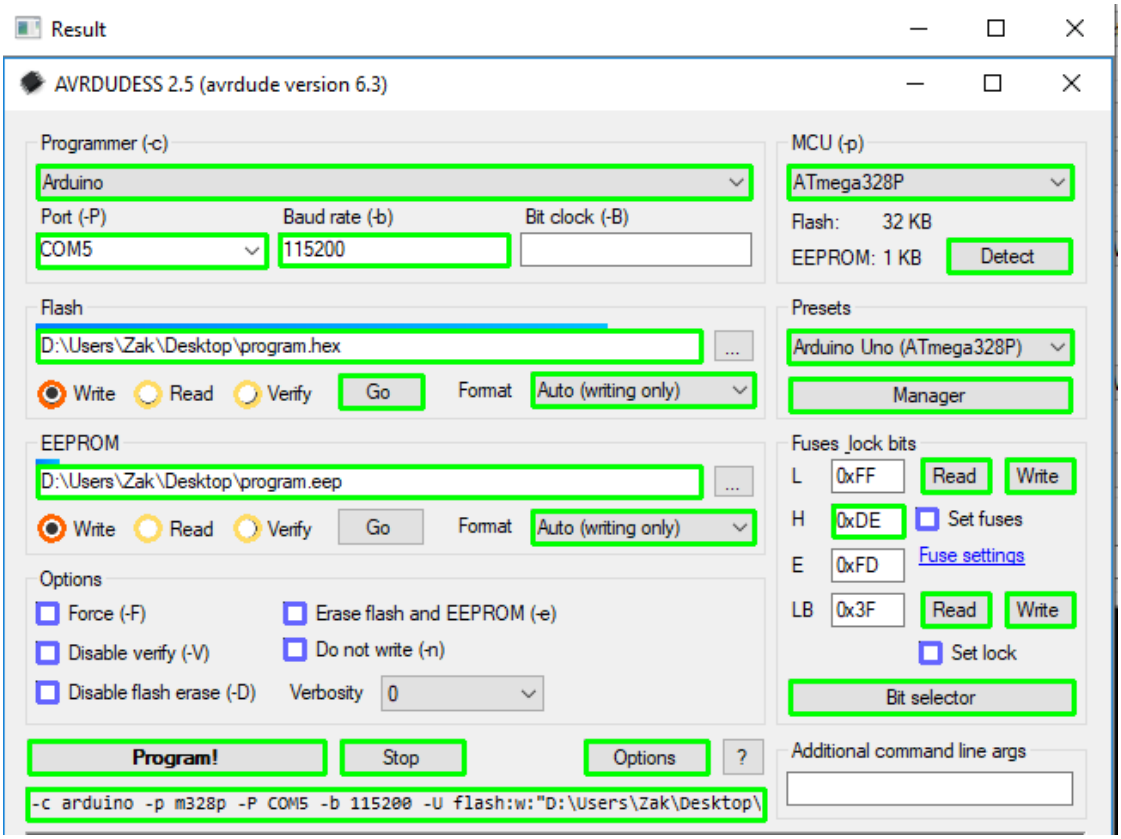


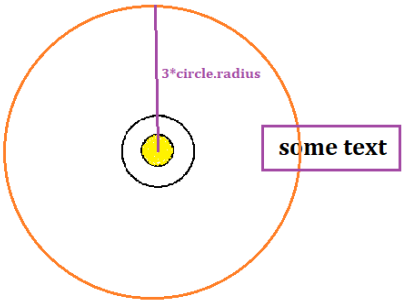

**Fig. 3.4. Detectarea check-boxurilor (albastru)**

## 3.5. Detectarea Radial-box-urilor

Pentru detectarea check-box-urilor se folosesc rezultatele obținute din funcțiile anterioare (detectarea dreptunghiurilor din contururi și detectarea textului).

Pașii pentru detectarea check-box-urilor :

1) Detectarea cercurilor folosing metoda din OpenCV **HoughCircles** (mai întâi imaginea originală este transformată într-o imagine Gray, ulterior ea este blurată cu filtrul median pentru a înlătura zgomotul).
2) Impunerea condiției ca ordonata centrului cercului să fie mai mare ca ordonata dreptunghiului de text (circle.center.y > rect.y)
3) Punctul dreptunghiului de text să fie în interiorului cercului determinat de central radial-button-ului și raza egală cu 3 * raza radial button-ului
4) Determinarea state-ul check-box-ului (checked, unchecked) prin aplicarea metodei de binarizare **OTSU** și verificarea ca pixelii de culoarea neagră să reprezinte mai mult de 25 % din aria check-box-ului (funcționează pentru intensitate mai închisă a bifului față de background)



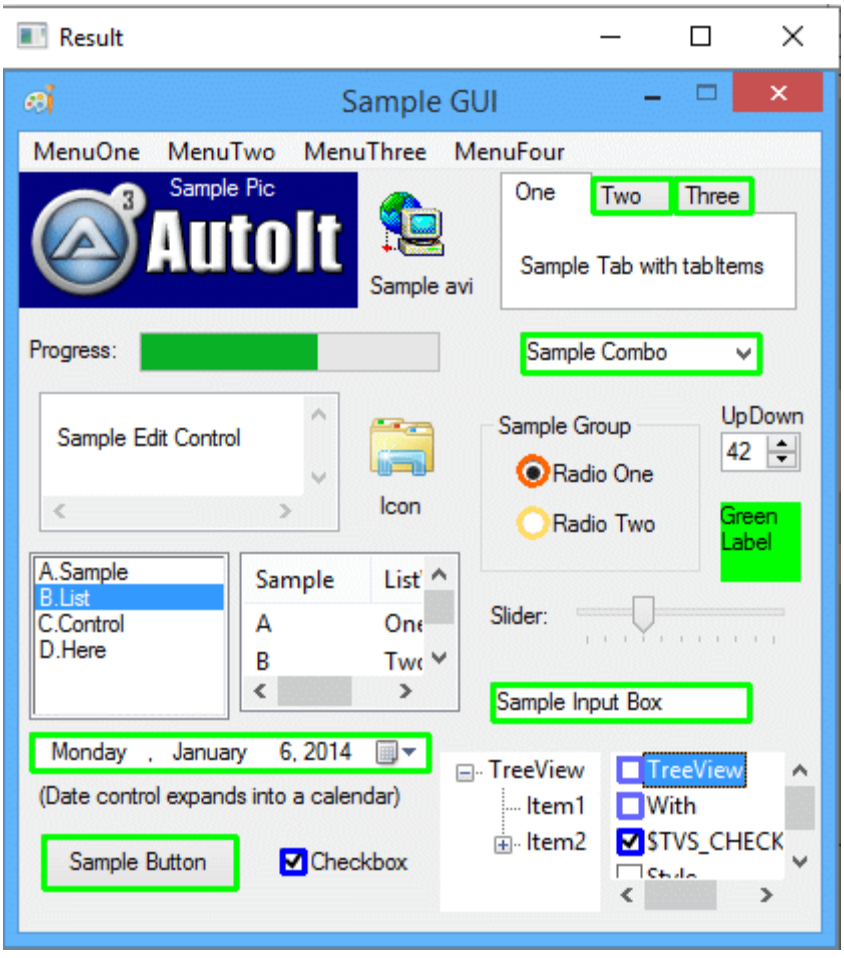




**Fig. 3.5. Detectarea radial-button-urilor (portocaliu)**

# Capitolul 4. Rezultate Eșuate

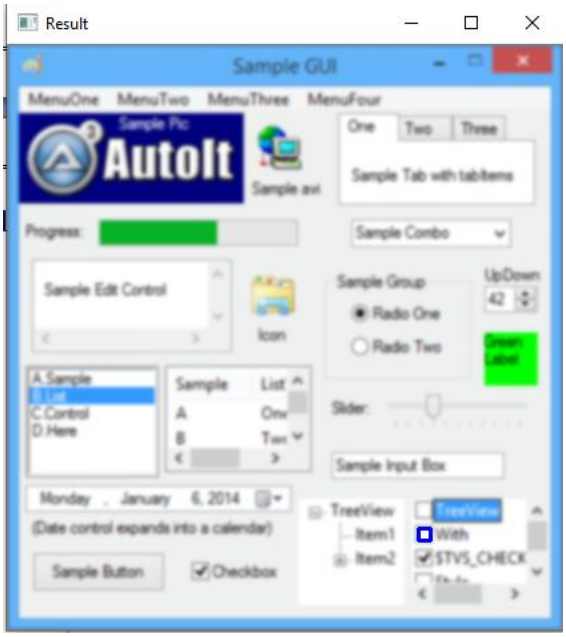



Fig 4.1

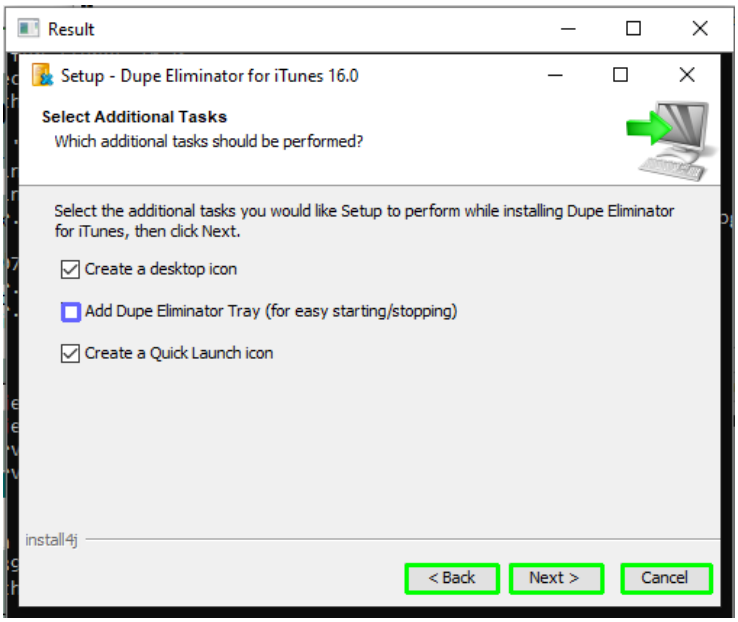



Fig 4.2

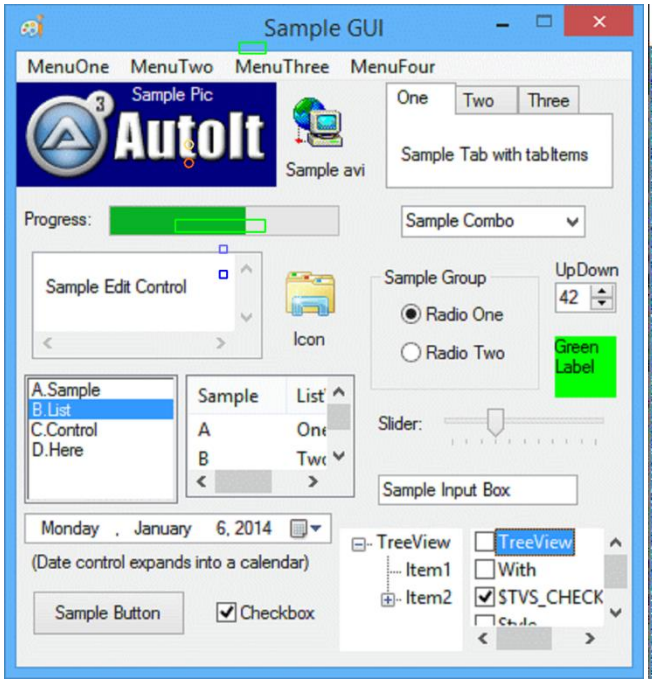



Fig 4.3

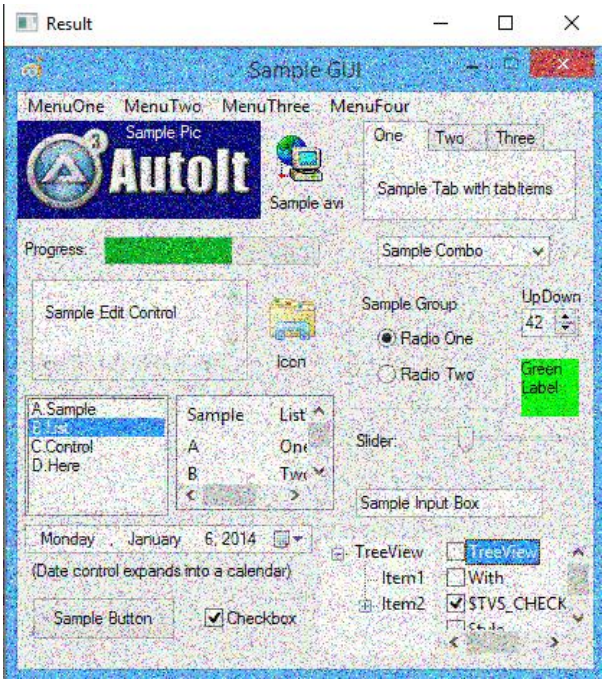



Fig 4.4

Programul nu detectează elementele grafice în cazul în care este blurată (Fig 4.1), mărită (Fig 4.3) sau conține zgomot (Fig 4.4)

De asemenea, nu s-a găsit vreo combinație de parametrii pentru funcția de detecție a dreptunghiurilor care să detecteze dreptunghiurile check-boxurilor bifate în interfețele grafice.

# Capitolul 5. Informații suplimentare

Rezultatele detecțiilor sunt puse într-un fișier de tip json, iar la vizualizarea rezultatelor ele sunt preluate din acest fișier și afișate.

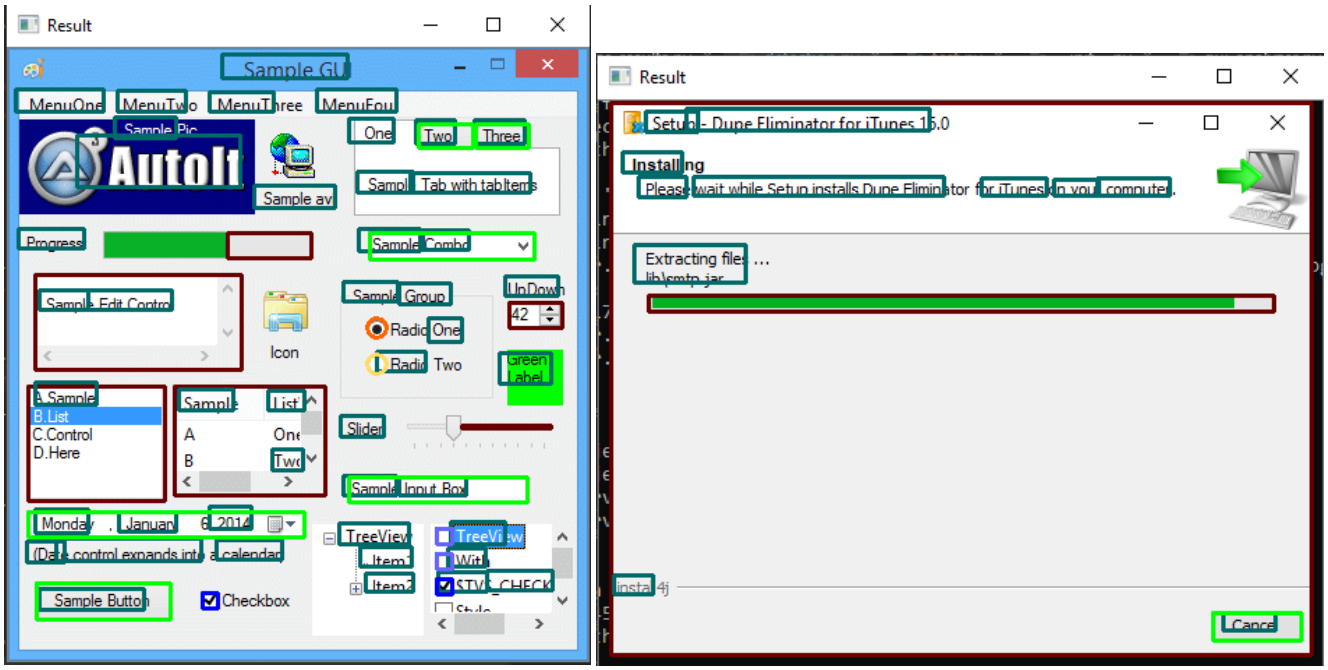



**Fig 5.1. Exemple de detecție text folosind OCR**


**Fig 5.2. Exemple de detecție text stocată în json (debug.json –** stânga**, results.json -** dreapta**)**

# Concluzii

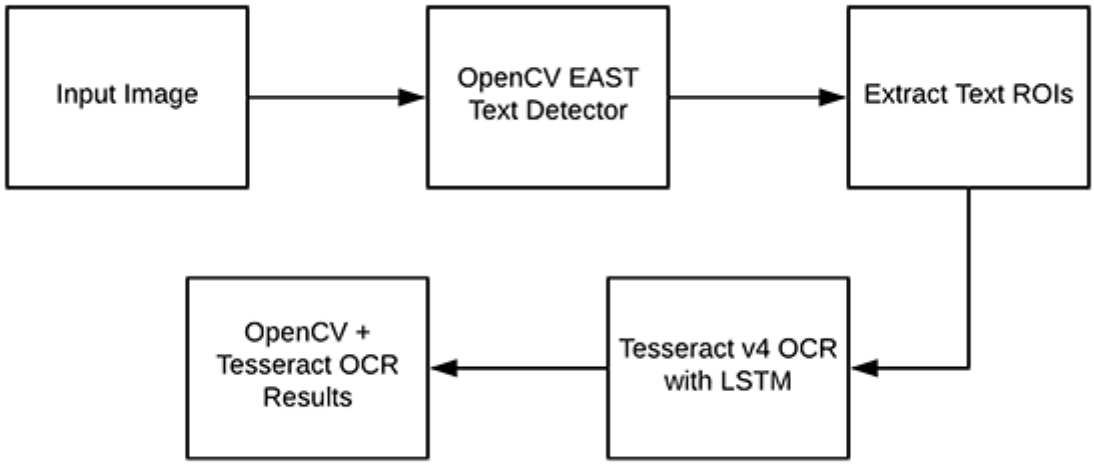



**Fig 6.1. Pipeline-ul fazei de detecție a textului**

Detectarea componentelor grafice nu poate fi realizată întotdeauna, programul implementat a funcționat bine în unele imagini, a eșuat în altele.

Unul din factorii care a cauzat acest lucru este datorat rețelei neuronale de detecție a regiunilor de text EAST și a motorului de detecție a textului Tesseract. Așteptarea ca OCR-ul să detecteze textul cu o acuratețe de 100% este pur și simplu nerealistă, unul dintre motivele principale fiind că modelul nu este instruit pentru anumite tipuri de fonturi de text.

Un al doilea factor care împiedică detecția precisă a componentelor este contrastul scăzut la unele componente grafice, precum și multitudinea de tipuri de componente grafice (ex. unele checkbox-uri/radial buttons pot avea regiunea check-ului de o intensitate mai mică ca cea a background-ului ceea ce ar împiedica detectarea check-ului).

O altă observație este că programul este sensibil la zgomot. O imagine blurată, mărită sau cu zgomot împiedică detecția elementelor grafice.

## Referințe


- "Extraction and Classification of User Interface Components from an Image" Saad Hassan, Manan Arya , Ujjwal Bhardwaj ,Silica Kole
- https://docs.opencv.org/master/d9/df8/tutorial_root.html
- "OpenCV Text Detection (EAST text detector)"
- "OpenCV OCR and text recognition with Tesseract" by Adrian Rosebrock
- BlobFromImage OpenCV Function
- Deep learning: How OpenCV's blobFromImage works